FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Player Tracking using Ultrawideband

João Miguel Bagoim Guimarães Ferreira Bernardo

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Paulo José Cerqueira Gomes da Costa

August 22, 2017

Player Tracking using Ultrawideband

João Miguel Bagoim Guimarães Ferreira Bernardo

Mestrado Integrado em Engenharia Informática e Computação

August 22, 2017

Abstract

Nowadays player detection, labelling and tracking in a field is done by using cameras and video analysis or by GPS-based systems to give managers and viewers statistics about each player. But those approaches aren't always viable.

This thesis is about building a system using time-of-flight radio transmitters (DWM1000 Module by decaWave will be used for this) to give a more precise estimation for the position of each player at any given point in time. Part of this task involves the creation of firmware for the communication between each transmitter and an information system to collect the data and process it for better understanding of each individual performance in the field. The purpose of this is to create a more accessible tracking system to be used in training or official games as some of the alternatives are either too expensive or too complicated to use.

To validate the reliability of such a system first some tests with a measurement tape will be made to confirm and adjust (if necessary,) the algorithm that calculates the distances to a single player. After that some calibration tests will be made to ascertain the system's ability to measure the field's limits.

Resumo

Atualmente, a detecção, identificação e localização de jogadores dentro de um campo é feita através de câmeras de vídeo ou por sistemas baseados em GPS, para produzir estatísticas sobre cada jogador para os analistas ou telespectadores. Mas essas abordagens nem sempre são viáveis.

Esta tese é sobre a construção de um sistema usando transmissores de rádio baseados em tecnologia de tempo de voo (módulos DWM1000 pela decaWave serão usados para isso) para dar uma estimativa mais precisa para a posição de cada jogador em qualquer momento dentro de campo. Parte dessa tarefa envolve a criação de firmware para a comunicação entre cada transmissor e um sistema de informação para recolher os dados e processá-los para uma melhor compreensão de cada desempenho individual no campo. O objetivo disso é criar um sistema de localização mais acessível para ser usado em treinos ou em jogos oficiais, porque algumas das alternativas são mais caras ou complicadas de usar.

Para validar a confiança no sistema, primeiro alguns testes com uma fita métrica serão realizados para confirmar, e ajustar (se necessário), o algoritmo que calcula as distâncias para um único jogador. Depois disso alguns testes de calibração serão realizados para averiguar a capacidade do sistema de medir os limites do campo.

Acknowledgements

The dissertation is the final step in a long road that could not have been done without the help of a few people.

I would like to thank my supervisor, Paulo Costa, for the help and guidance that helped me broaden my knowledge about things i didn't know before.

I would like to thank my stepfather for all that he has taught me throughout the years that helped me improve as a programmer.

I would like to thank my mother and father for supporting me in my life and always being there when i needed them.

I would like to thank my grandparents and aunts for always being there when i was growing up and made me the person i am today.

Lastly i would like to thank my brothers and sisters for always managing to brighten my days even without knowing they were doing so.

João Bernardo

"All that is gold does not glitter,
Not all those who wander are lost;
The old that is strong does not wither,
Deep roots are not reached by the frost.
From the ashes a fire shall be woken,
A light from the shadows shall spring;
Renewed shall be blade that was broken,
The crownless again shall be king."

J. R. R. Tolkien, The Fellowship of the Ring

Contents

CONTENTS

List of Figures

LIST OF FIGURES

List of Tables

Abbreviations

API	Application Programming Interface
CSV	Comma-Separated Values
GUI	Graphical User Interface
GPS	Global Positioning System
IDE	Integrated Development Environment
MVC	Model-View-Controller
NFL	National Football League
PCB	Printed Circuit Board
SPI	Serial Peripheral Interface Bus
UWB	Ultra-Wideband
TDOA	Time Difference of Arrival
WI-FI	Wireless Fidelity

Chapter 1

Introduction

1.1 Motivation and Context

Sports is all around us and the investments made around them has never been so large, with entire broadcasting channels dedicated to them, and to the analysis of every single detail throughout the matches, to provide the public with every information they might need or want about the game they most love.

Teams also felt the need of having analysts in their staff that provide coaches the most accurate information possible about their opponents and their own players in order to create the best tactical plan that gives the best chances of winning.

In order for coaches to have access to that information the analysts had to automate the process of gathering the information so that the training schedule could be optimised for everyone involved in the team.

Current tracking systems may be too expensive for teams with less monetary resources. For that reason developing a cheaper alternative would provide more teams with the ability to better analyse their games and practices.

1.2 Dissertation Structure

Other than the introduction this dissertation has five more chapters describing the work accomplished.

In chapter 2 a study of different existing methods of tracking players is given as well as an alternative radio communication transmitter to the one used in this dissertation.

In chapter 3 the results of the first reliability tests using the DW1000 module[DWM] are shown to give a better understanding on how it works and how the player tracking will work with it.

In chapter 4 it is described all the software development required for the player tracking system to function and how each player position is calculated in a given sport's field without the need for previous manual measurements of the field.

In chapter 5 the results of the system performing in different conditions are show in order to assess its reliability.

Finally in chapter 6 a conclusion of the overall system is described as well as future improvements to the system to make it sturdier and give better results.

Chapter 2

State of the Art

The use of video cameras in player detection and tracking for statistical analysis is the most used nowadays but it is not fault proof and can't be applied to all sports easily.

As such there is a need to develop a better system that ensures more accuracy in the values retrieved and that can be applied to a broader group of sports.

2.1 Player Tracking

2.1.1 Using Video Analysis

Most of the player detection in sports is done using image processing software to properly identify each individual present in the playing field.

There are several approaches on how to do it, but the general idea is by having a learning system to properly recognize the individuals, and then using previously collected samples, in the field and then group them in the correct team or mark them as the referee. The system is then able to track the players, using the help of a prediction algorithm in order to correct any mistakes that might occur.

One of the disadvantages of this sort of system is that it is not 100% reliable [LTL+09] due to the misinformation retrieved from analysing some of the frames in the video. That misinformation can be caused by several reasons such as player occlusion, blurred frames or the algorithm mislabelling one of the individuals.

Another disadvantage is the cost attached to the purchase of several cameras and the need to position them at a high enough position in order for them to be able to view the whole playing field and the players on it which for some cases like a practice field, it is impractical.

2.1.2 Using Trilateration

This method of player detection require the player to use some sort of device that would help locate them in the match and then use reference devices to use a process called trilateration to determine their position.

Unlike triangulation, which uses angles to calculate the position of a certain point, trilateration employs the distance between an unknown located point and three or more well known points to discover its absolute position utilizing the geometry of circles or triangles (2D) or spheres (3D) for it.

A common use of this method is GPS which works well outdoor for tracking people or objects that are either stationary or slowly moving but when it comes to tracking people running and making sharp and quick turns like it happens in most sports, it has an error margin that is not acceptable. It also does not work indoors where sports like basketball or futsal are more commonly located.

2.1.3 Other uses

Other concepts have been developed that mimic GPS but for indoor environments that could be used as an alternative to video cameras.

One of those uses is based on the WI-FI signal [THU12] where the position of the Wi-Fi device is calculated based on the signal strength retrieved in the access points used as references. For it to work the system must be trained first with known distances so it can translate the signal strength to a distance in order to calculate a position.

Some drawbacks that can affect the strength values are the orientation of the device as well as any human body covering part of the signal path. Other problems are related to the amount of noise that needs to be filtered in order to receive the correct signal and a prediction algorithm to calculate the position outside the observation points.

Bluetooth works in a similar way to Wi-Fi but adds the additional problem of pairing the devices and the longer it would take to establish connections and avoid all the noise in the air.

Such problems affect the accuracy of a system based on Wi-Fi or Bluetooth and are not a good alternative to video cameras.

The NFL is currently using a technology based in UWB by a company called Zebra [Dar] that utilizes tags in the players shoulder pads (Figure 2.1) and then spreads sensors around the field that detects the pulses sent by the tags and then communicates that information to a central system that will compile the data for the analysts to use. The accuracy of this system is around 15 centimetres.

2.1.4 Position Calculation

In [LTLM13] the authors are utilizing a system based on the same DW1000 chip used in this dissertation to track the *"positioning of a wheeled robot on a production floor inside a factory"*.

Figure 2.1: Player shoulder pads with Zebra's RFID tags

The method includes the placement of several anchors, at the same height around the place the robot will be moving, in a triangular disposition show in figure 2.2.

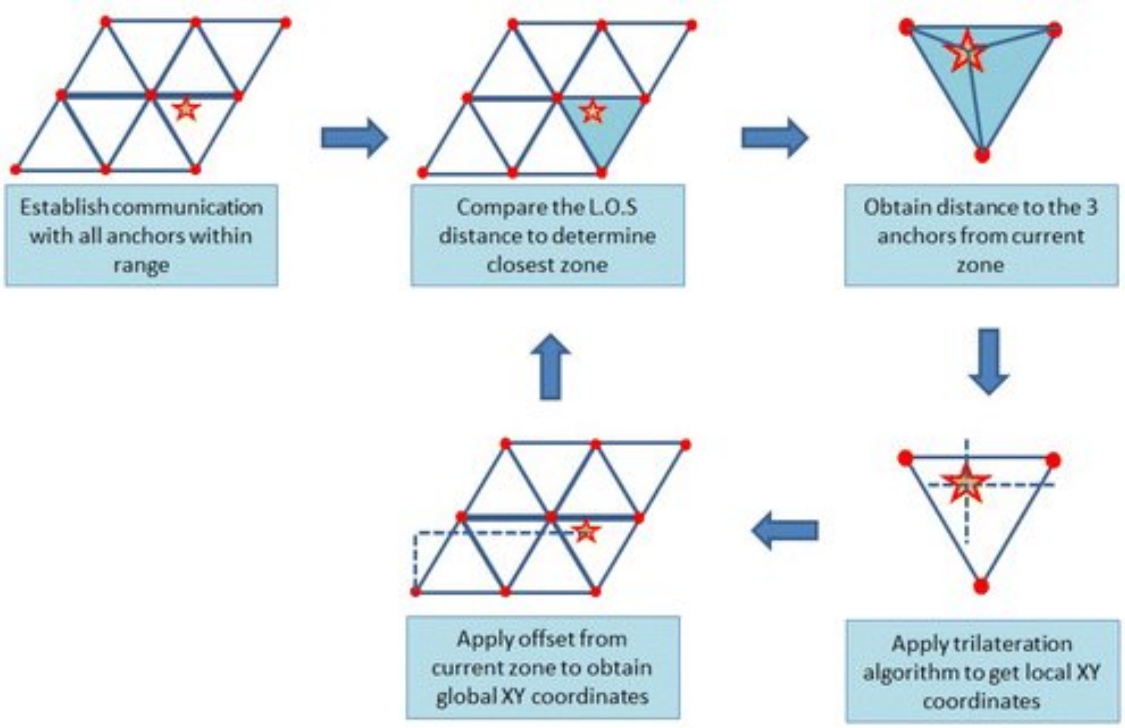


Figure 2.2: Triangular disposition, from [LTLM13]

This triangular disposition allows for the system to work across a large area where the accuracy of the transmitters starts to decrease as it calculates the position of the tracker related to the anchors that provided the distances and then add an offset depending on which anchors were used.

The trilateration algorithm they used is based on the intersection point of three spheres (figure 2.3). Since the positioning of each anchor is known as well as their distances to the tracker, a system of equations was used to discover the unknown coordinates of the tracker's position.

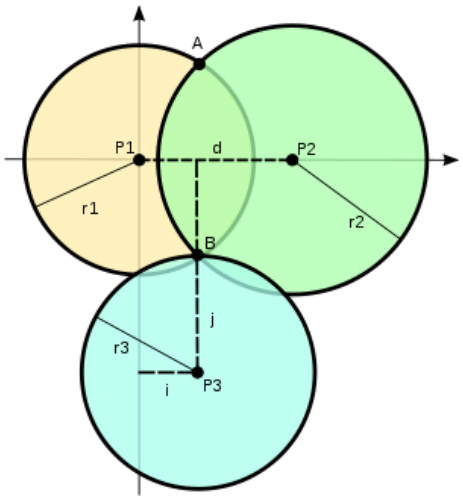
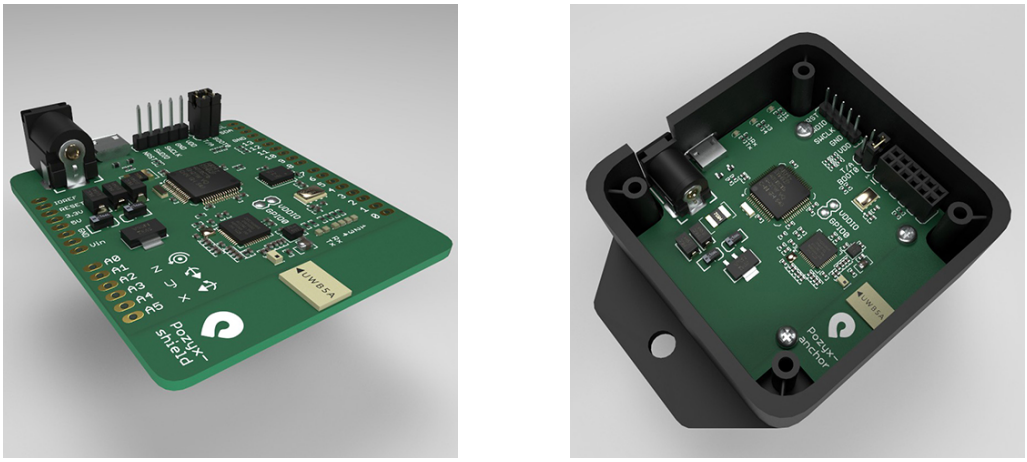


Figure 2.3: 2D representation of Trilateration

2.2 Hardware

An alternative to the localino devices[HL] that were chosen for this dissertation are the devices created by [Poz] and that can be viewed in figure 2.4.



(a) Pozyx tag

(b) Pozyx anchor

Figure 2.4: Pozyx devices

Both the devices represented in figure 2.4 contain a DW1000 module for the radio transmission and a micro controller to process the information coming from it and communicate it to the exterior. The main difference from the localino devices is that the pozyx tags also contain motion sensors to provide extra information about the tag's status and movement. The informations retrieved from the sensors can determine the tag's orientation, angular and linear acceleration, gravity among others.

The pozyx also has two libraries to help in the development of a positioning system. One written in C++ for the Arduino and another in Python to interact with the different devices.

2.3 Development Tools

Having only the communication between the different devices and the calculations required to determine their position would just lead to large amounts of information and deciphering what was happening would be difficult. With that in mind some research was made on different frameworks in order to create a GUI that would display the events in real time and provide the user with the ability to understand what is happening in the system.

Due to previous experiences, the first framework researched was Swing[Swi]. Swing is a GUI framework written in and for Java development thus providing platform independence and removing the need for modifications to allow the applications to run in any operating system. It is very versatile because the interface is made from different components that can be extended and customized depending on the development. Plus, Swing follows a MVC design pattern which provides a level of abstraction between the graphic representation of the components and the handling of data behind it.

Another framework researched was Lazarus [Laz]. Unlike Swing (that is a library), Lazarus is a visual IDE that allows the development of both graphical or console applications. Lazarus uses the Free Pascal compiler that supports different dialects of the programming language Pascal and is able to compile applications that run on any system where a Free Pascal compiler exists. It includes a windows layout designer to help in the creation of the graphical interfaces.

The last assessed framework was [QT]. QT is a C++/Python framework for creating cross-platform GUI. Paired with the QT Creator IDE it allows for a quick creation of an interface with its QT Designer tool that assists in the composition and customization of windows. QT also has a Signals and Slots communication system between objects that leads to an easy way to update the information in different objects when an action occurs in the interface.

2.4 Conclusions

Player tracking through the aid of video cameras is the most widely used tool in sports but it requires a lot of system training and still some errors can come from misanalysed frames and as such having a system that is capable of more easily distinguish players without the need of a training phase could provide beneficial in today's world.

Even though there has been some development in the area of indoor real time location systems, that may also be used outside, those processes are only able to track immobile or slow moving people/objects with predictable walking patterns in order to track their position at any given point in time. The accuracy of such systems is still far away for the use in sports where there are fast moving people most of the time and the calculations must have the least amount of error possible.

Chapter 3

Distance Measurement

3.1 Proposed Solution

The solution proposed in this paper is to create an alternative to player tracking during the course of a match or a practice and it was started in Microprocessor Based Systems class. The first tests were made utilizing two systems described in figure 3.2 in order to discover the best hardware and software configurations according to the datasheet specifications of the UWB module [DWM] to produce the best results possible.

The communication between both systems was analysed to find any errors in the distance measured so that it could be corrected for a further development of this technology.

3.1.1 Ultra Wideband

Like the name says, UWB is a type of radio transmission that uses short pulses over a larger bandwidth of frequencies, rather than using a specific frequency, to transmit information at a low power usage. It is usually used at frequencies between 3.1 and 10.6 GHz where the bandwidth is at least 20% of the transmission frequency value in order to avoid interference with already existing radio transmission systems.

3.2 Hardware

The hardware used for the tests was the DWM1000 connected via SPI to an Arduino NANO. The Arduino was chosen due to its popularity and its cost, making it the most viable choice for this sort of problem.

The DWM1000 implements the standard IEEE 802.15.4-2011 that is related to Low-Rate Wireless Personal Area Networks (LR-WPANs)[Hä16]. For that reason it has an internal clock running at a frequency of 63.8976 GHz and a 48 bit counter which enables it to easily measure time in picoseconds. It registers the time the messages are being received as well as setting a time for when a certain message should be sent and has an extreme accuracy on doing so.

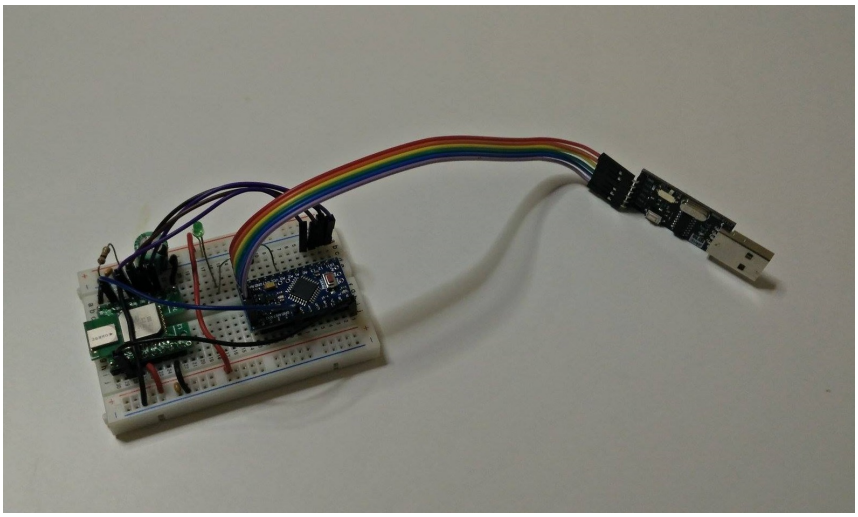


Figure 3.1: Utilized System

The Arduino is connected to the computer via serial port for communication and display of results gathered between the communication of the two modules using during the tests.

Two circuits like the one shown in figure 3.1 where utilized in this part of the development.

3.3 Software

The code used in these initials tests was the one available at [Tro] which facilitated the complex work of configuring the communication between the Arduino and the DWM1000. The library provided there will be used throughout the rest of the project.

The most basic process to measure the distance between two devices is represented in figure 3.2. Device A sends a message and then waits for device B to reply with the value of the time it took since it received the message until it sent the reply (*Treply*). As the speed of the radio waves is the speed of light we can calculate the distance that one device is from the other.

According to the DWM1000 manual, this method shouldn't be used as the degree of error involved is too big for it to be useful. Instead a Double-Sided Two-Way ranging method should be used. It works by performing the previous process twice but in the second time the roles are reversed with device B starting the transmission.

This method is shown in figure 3.3 is a version of the method that uses only three messages instead of four when linking the two processes together. This chaining is done in an asynchronous way which helps in calculating distances, since in a synchronous way we would have one less *Tprop* to adjust any error that may appear in the measurements.

By having the times each transmission was either received or sent, the values in 3.1 can be filled in order to calculate the time the messages took to arrive from device A to device B and with that the actual distance between them.

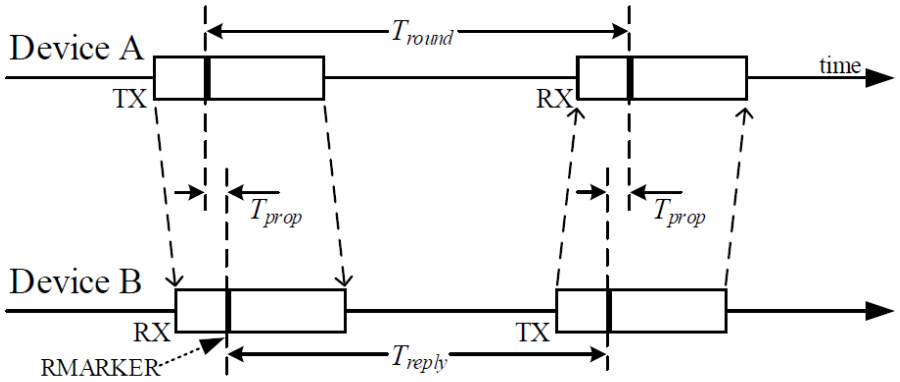


Figure 3.2: One-Sided ranging

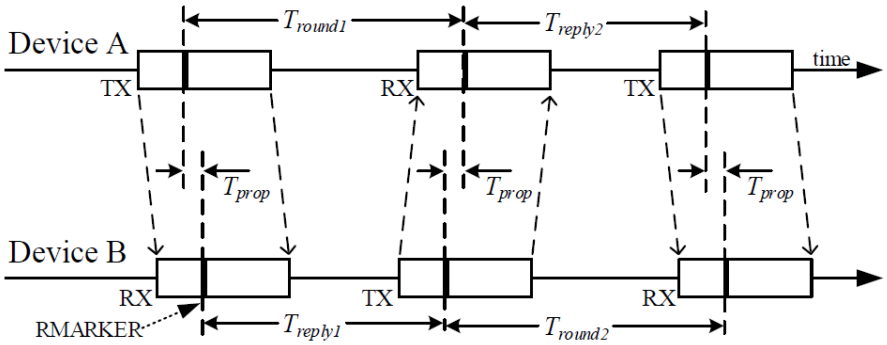


Figure 3.3: Double-Sided Two-Way ranging

$$Tprop = \frac{Tround1 * Tround2 - Treply1 * Treply2}{Tround1 + Tround2 + Treply1 + Treply2} \tag{3.1}$$

3.4 Results

To measure the error that might exist in the systems or the software some measurements were made at known distances and then compared to the results obtained in order to calculate such error.

As it is visible in the values obtained in figure 3.1 and in the resulting graph in figure 3.4 the results are different than the real ones but they follow a linear progression which means the correction of that error can be achieve with relative ease in order to normalise the distances.

Table 3.1: Distance Measurement Results

Real distance (meters)	Minimal Distance (meters)	Maximal Distance (meters)
0.08	0.06	0.10
0.50	0.41	0.46
0.80	0.92	1.00
1.00	1.19	1.23
1.25	1.50	1.59
1.57	1.93	2.00
1.80	2.38	2.43
2.00	2.47	2.53

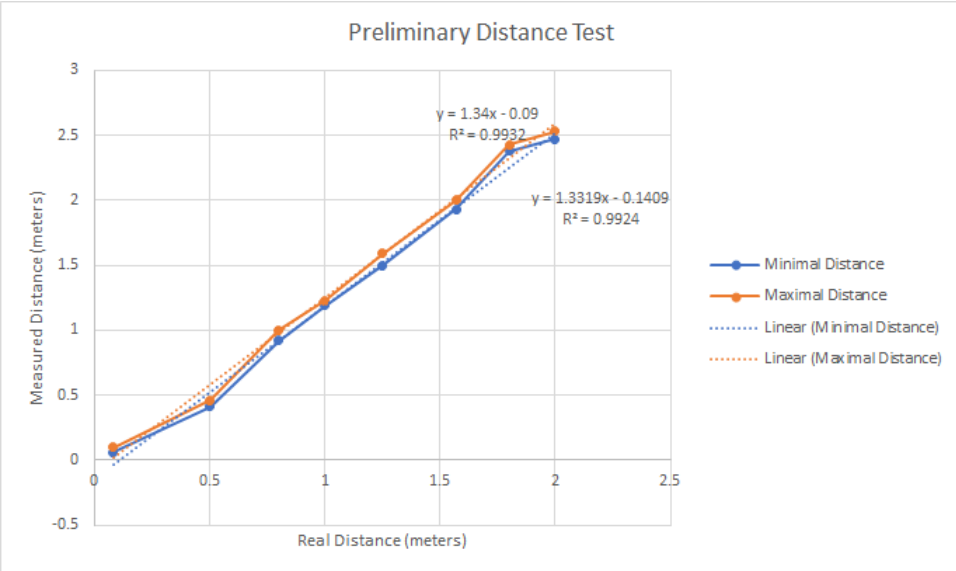


Figure 3.4: Graph Results

Chapter 4

Implementation

In order to create the tracking system described before there are two distinct areas to be developed to achieve a working solution. First there is the firmware implemented onto the microprocessors that handles the transmission of messages between three anchors and multiple tags. This firmware will be able to calculate the distances between the different devices with TDOA.

To display the positions of the devices a secondary software was designed that will be able to receive the different distances and turn them into a real position in a 2D representation of the game field where the tags will be moving.

The system that will be used in this dissertation consists of a computer and six radio communication devices. The communication devices will be divided in two where one half will be treated as anchors and the other half as tags. The anchors remain stationary, in key spread out positions around a field, while the tags will be freely moving and attached to the players to track their positions. These devices will handle all the communication details as explained in section 4.2 and the treatment of the messages traded between them needed for distance calculation. One of the anchors is attached to the computer to collect the messages shared between all the devices and provide the user with their visual representation. It will also be able to receive commands from the computer to update its settings and behaviour.

4.1 Hardware

The first hardware used for this problem was big and mainly for testing purposes in order to figure out the viability of the project and what adjustments had to be made in order to work.

As the devices are supposed to be worn by players, a more compact solution had to be found so that it wouldn't affect the players more than required. As such there was a need to find such a solution.

The first experiment was with a PCB developed by [Way] that contained the connections needed between the microprocessor and the radio transmission device and can be seen in figure

4.1. Having the PCBs all was left to do was to solder an Arduino Pro Micro and the DWM1000 on it and then test it.

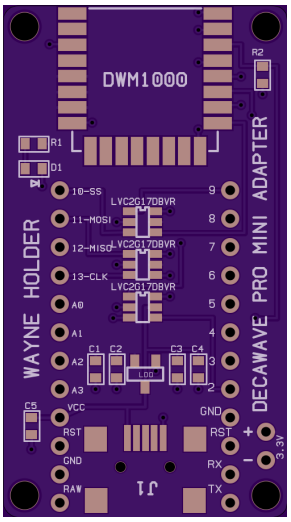


Figure 4.1: Wayne's PCB

While testing this solution there were some communication problems between the Arduino and the DWM1000 resulting from the poor quality of the board that didn't exist in the original solution tested and as such had to be abandoned and another solution was needed.

The second experiment was more successful and it is the one used throughout the rest of the project. It was developed by [HL] and offers a smaller solution than the previous one because, instead of needing to solder an Arduino on top of it, the same processor used by the Arduino is already part of the board figure 4.2.

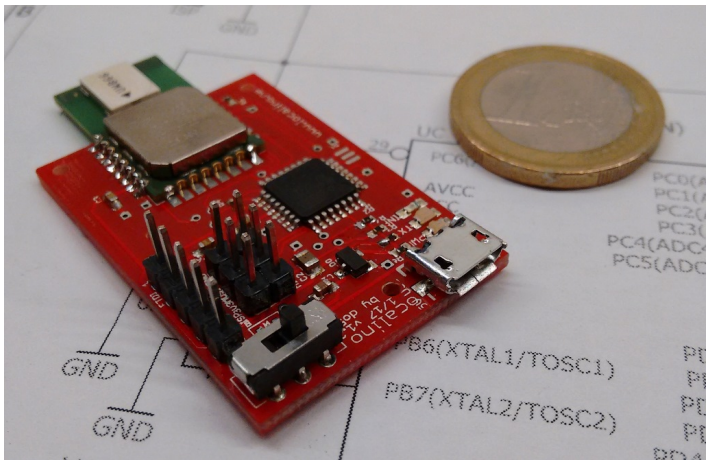


Figure 4.2: Localino

4.2 Microprocessor Firmware

Resuming the system description in chapter 3 it was decided to alter the existing code for one anchor and one tracker and expand it to be able to handle multiple anchors and trackers.

The main algorithm to achieve is described in figure 4.3 and it assesses the distances between three anchors and a single tag. It is aimed at reducing the number of messages needed to calculate the position of the tracker.

The way it works is that the tag sends a message saying it wants to start the transaction to all the anchors and then waits for the reply of each anchor. When the tag has received all the replies it returns the timings of all the messages sent and received so that the anchors can calculate their distances to the tag.

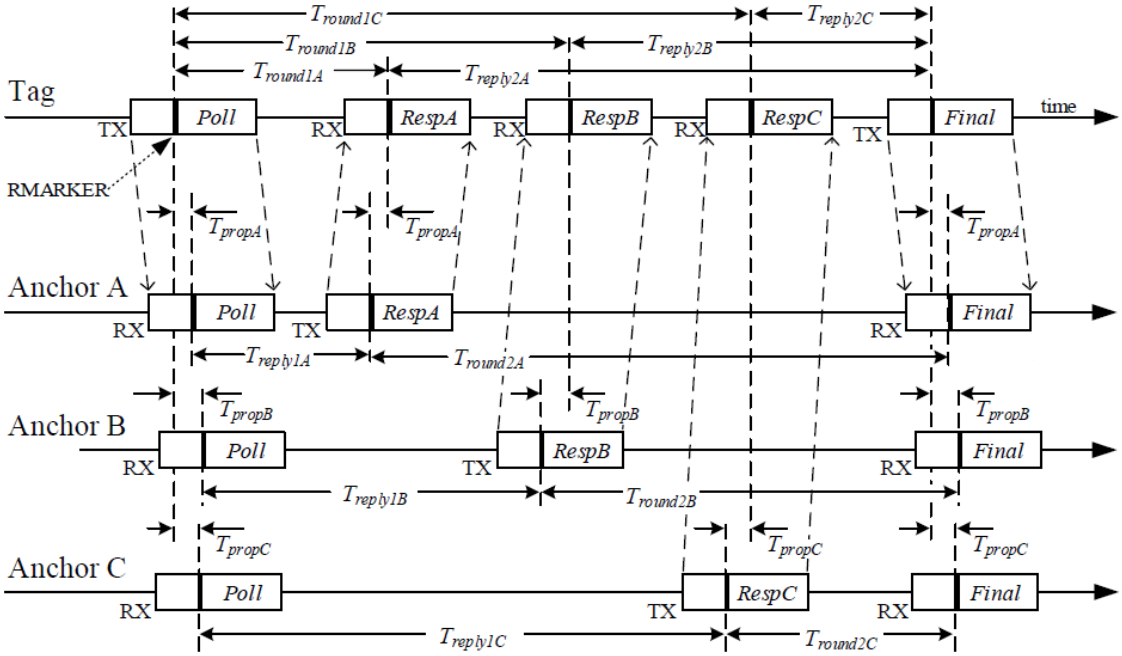


Figure 4.3: Three Anchors Double-Sided Two-Way Ranging

This algorithm was developed in the Arduino IDE in C programming language with the assistance of the library [Tro] on the communication between the microprocessor and the radio transmission device [DWM].

In order to give a clear understanding of the state machines represented in figure 4.4, 4.5 and figure 4.6 an explanation of the message types exchanged between the different devices is needed:

- INIT - A message sent from the anchor connected to the computer in order to start a round of communication between a tag and the anchors
- RANGE INIT - A message relayed from the anchor connected to the computer in order to start a round of calibration between an anchor and the remaining anchors

- POLL - A message sent from a single tag to all the anchors requesting their reply
- ACK - A response sent from the anchors to a POLL request
- RANGE - A message sent from a tag with its messages' received and sent timings to all the anchors
- RANGE REPORT - A message sent from the anchors to the main one with its distance to a tag so that all distances to a certain tag can be saved in the computer

Other than the INIT and RANGE INIT messages that contain an id to the device that will be starting the communication round, all others will also contain the id of the device that is sending the message so that missing messages from a tag or an anchor can more easily be detected.

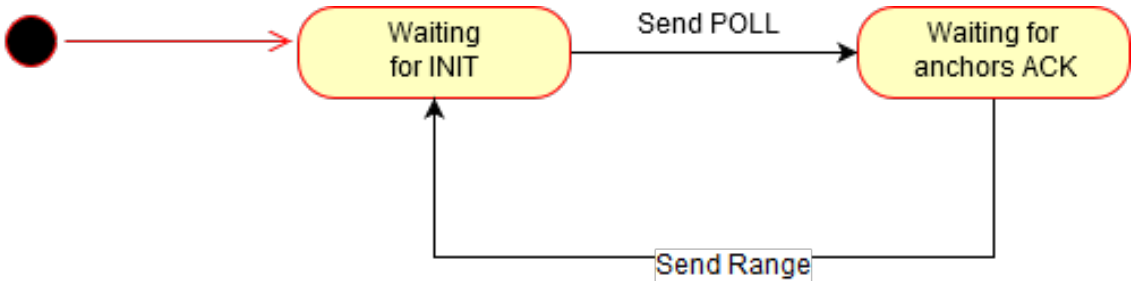


Figure 4.4: Tag State-Machine

In figure 4.4 we can visualize the state machine referring to what is occurring in the devices set as tags. The default state is the one where it is waiting for a INIT message from the main anchor. In order to avoid the overlapping of messages the INIT message also contains an id of the tag that is suppose to start that round of communication. The other tags also read the message but as it is not directed at them they ignore it and wait for one with their own id.

The tag that the INIT message is referred to sends a POLL message to all the anchors and saves the time at which that message was sent. It then waits for all the anchors to reply to that message with an ACK message of their own and saves the time where each message arrived. Finally returns those times in a RANGE message and the expected time the RANGE message will be sent and that will give the anchors the ability to calculate their distance to the tag.

In figure 4.5 we can visualize the state machine referring to the normal behaviour of the devices when they are acting as anchors. Like the tags they wait for an INIT message in order to know when to listen for a POLL message from the tag referred in the INIT message.

After the anchor receives a POLL message it will wait for a time, in microseconds, calculated based on its id (equation 4.1 where 3000 is the value that didn't produce any overlap), so that there isn't an overlap on the tag in receiving all ACK messages at the same time causing it to miss one

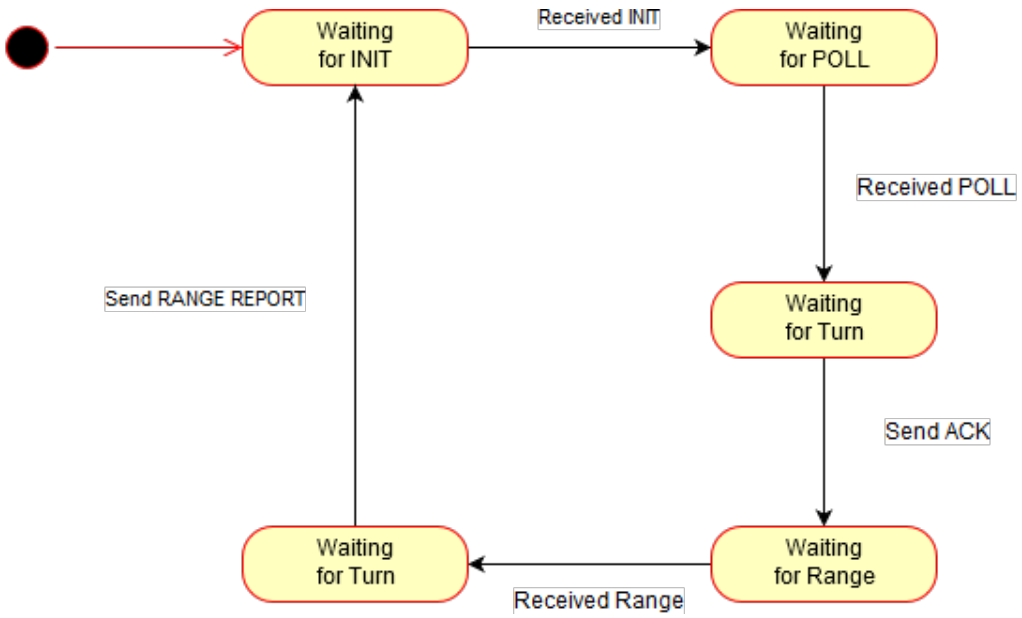


Figure 4.5: Anchor State-Machine

or more of those messages. When the ACK message is sent, the timing at which it was sent and the time where the POLL message was received are saved.

$$T_{wait} = AnchorId * 3000_{ms} \tag{4.1}$$

Once that is done, the anchor will wait for a RANGE message from the tag containing the times related to it so it can calculate the time it takes for a message to travel between the tag and the anchor. That time can once again be calculated with the equation in 3.1 in chapter 3. To get the variables present in 4.3 we do the following operations with the values both the tag and the anchor recorded during their communications:

$$Tround1 = Tackreceived - Tpollsent \tag{4.2}$$

$$Treply1 = Tacksent - Tpollreceived \tag{4.3}$$

$$Tround2 = Trangereceived - Tacksent \tag{4.4}$$

$$Treply2 = Trangesent - Tackreceived \tag{4.5}$$

where the equations 4.2 and 4.5 are based on the values retrieved from the tag's clock and the equations 4.3 and 4.4 are based on the values retrieved from the anchor's clock. This leads to the clocks not needing to be synchronized in order for the algorithm to work and give real distances between the devices with minimal error.

Although all anchors are running the same code, the one connected to the computer will also receive commands through the serial port that will then broadcast to the rest of the system what is

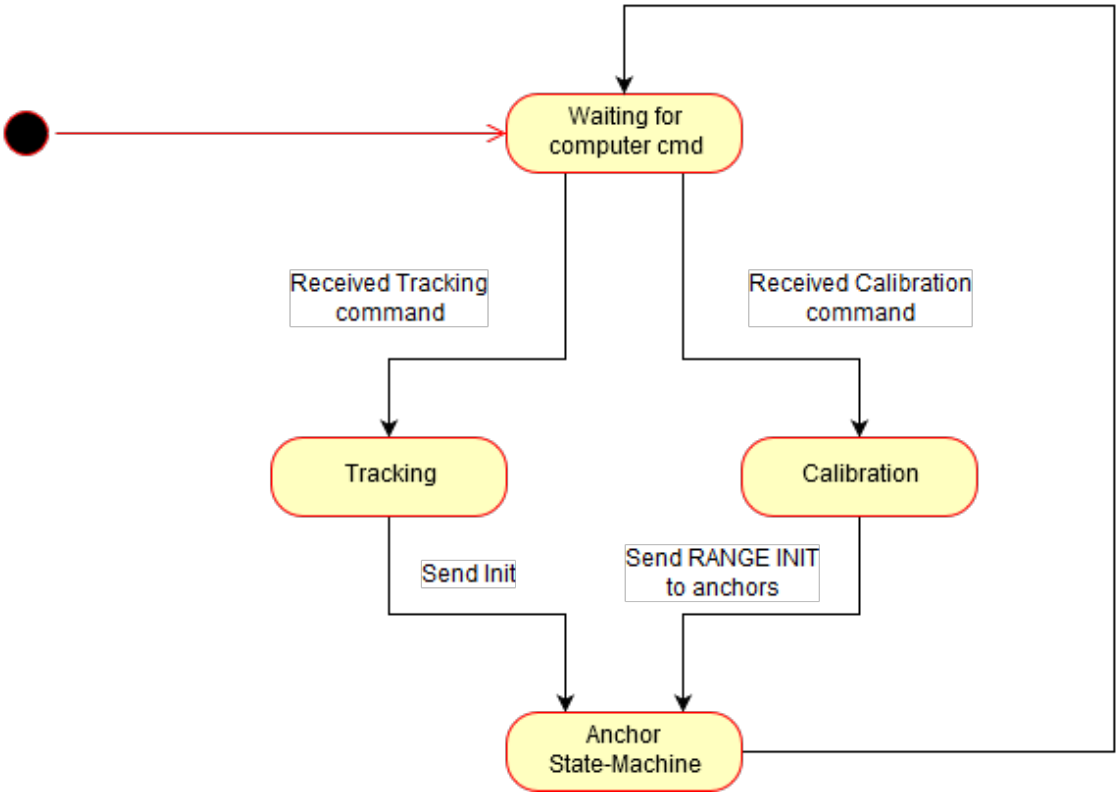


Figure 4.6: PC-connected Anchor State-Machine

suppose to be doing.

Two of the commands the main anchor will receive from the computer are based on the current setup of the system containing either the number of anchors or the number of tags present in the system. Those values will then be transmitted to the system either in the INIT or RANGE INIT messages so that every device can update their variables to the most current setup.

The other two commands are to define if the system will behave in calibration mode or in tracking mode as we can see in 4.6. The tracking mode is based on the interactions described in figure 4.4 and figure 4.5 and the main anchor behaves as a regular anchor through the process with only difference being that it is responsible for sending the INIT messages for each tag at the start. This is done to ensure that if for some reason an interaction with a tag may fail, the following interactions don't suffer from that. It also allows for the system to only track a specific tag if need be.

In the calibration mode a RANGE INIT message will be sent only for the anchors and it that message it will define an anchor to behave as a tag so that we can calculate the distance between the anchors in order to reach a real size of the field the tags will be moving on and its limits. More details on how this is done in section 4.3.

4.3 Visualization Software

Instead of having just the output of the devices with the different distances between the tags and the anchors a software was create so that the information could be better visualized.

But before that data can be visualized some configurations need to be selected so it can represent the information in the most accurate form possible. In figure 4.7 the user is present with the options to select the type of field the devices will be used on as well as the number of players in each team will be using trackers. There is also a field for a file name so that the recorded information can later be saved for later use in creating statistics about the game.

Figure 4.7: Configuration Window

If the number of players is set to a non-zero value the user is presented with a window (figure 4.8) where a name can be set for each individual player. Otherwise the name will be set to a generic Player x where x is the tag number the player was carrying.

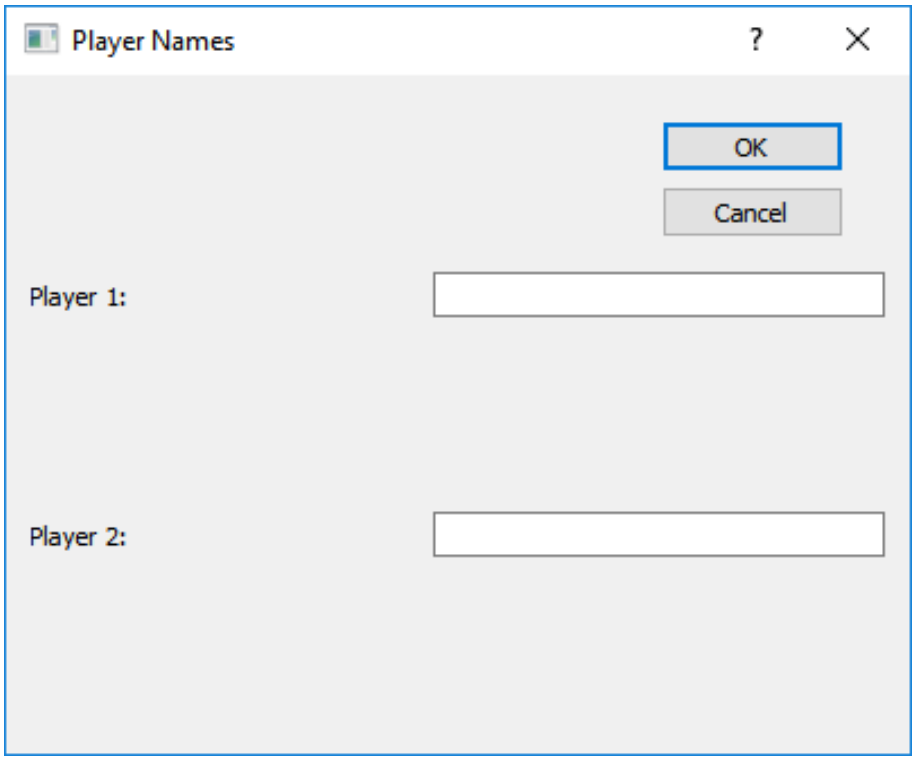


Figure 4.8: Player Naming Window

Another important part for the system to work is having the real measurements of the field and the position of the anchors around it so we can get a 2D representation of the trackers in the field. For that to work there is a calibration menu that allows to either load an existing calibration or create a new one if it is the first time.

To both calibrate the system and calculate the position of each player it was opted to use a C library [Wut] called lmfit that is based on Levenberg-Marquardt algorithm which is an algorithm to solve non-linear least squares problems by utilizing two minimization methods, Gradient Descent and Gauss-Newton [Gav13], in order to achieve a faster and more precise solution.

The parameters on the Levenberg-Marquardt algorithm are updated depending on the the last iteration where if the difference in results is large enough the change in parameters is based on the Gradient Descent method while small differences lead to an update using the Gauss-Newton method.

The way the algorithm works is that we give an estimative for where the point should be, and an evaluation function that calculates the error for that point. The evaluation function determines the error by calculating the distance from the given point to each anchor and then multiplying that value with itself. In each iteration of the algorithm a new point will be given to it in order to find the minimal error possible and the most likely position for the point.

During the calibration process (figure 4.9) several measurements are made. First to calculate the distances between each of the anchors that are placed around the field and then to three tags. Those tags are placed in specific places in the field so that after calibration process is done the

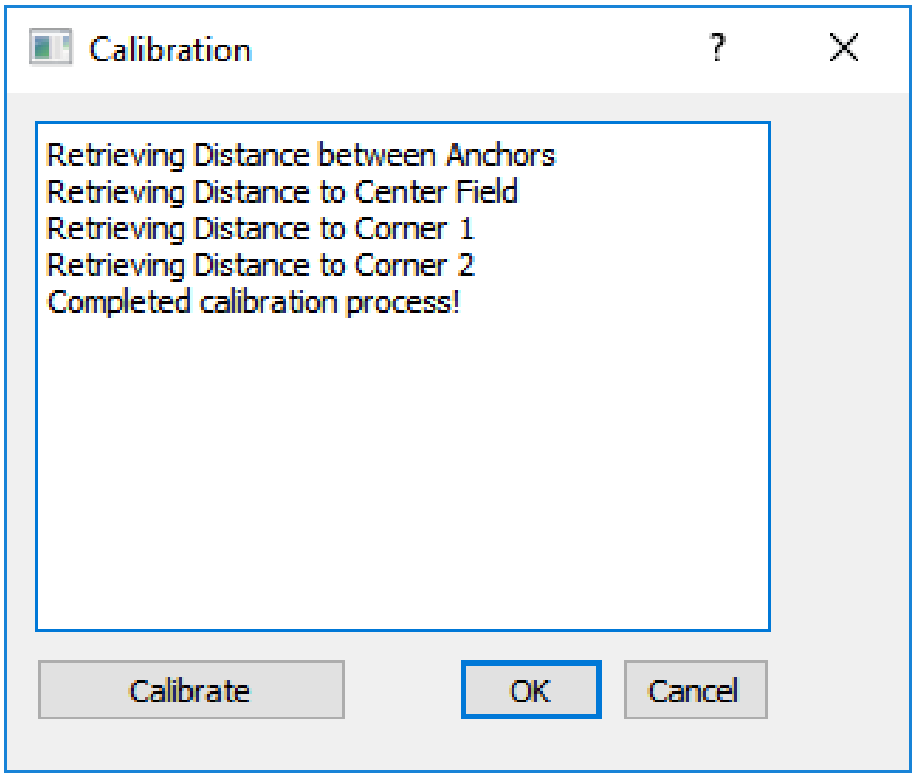


Figure 4.9: Calibration Window

player's movements in game are well represented on screen. One tag is placed in centre field so it's position can be assigned to the origin of an axis around which the remaining positions will be calculated. The other two tags are placed on two corners on the same side so their Y position is the same and the field's representation is faced to where it is required.

The system by default is collects distances for around two minutes in a rate of three cycles per second so the actual distance between each of the elements can be more precisely calculated by averaging all of the values and providing the system a more precise one. Around a third of the values collect are dropped out of the calculations in order to remove any outliers.

Once all the configuration and calibration is done the event (either a practice or a game) can be started and the players positions will start appearing on the screen.

To calculate the position of a certain player we have two variables we want to discover (X and Y position of a tag in a playing field) and we have the distances the tag is from each anchor and the positions of the anchors. The problem is that the distances to the anchors aren't perfect so we can't be sure there will be an overlap between each of them, otherwise the solution to the problem would be much simpler as the tracker's location would be the intersection of circumferences centred on each of the anchors with the radius being their distance to the tag.

In Figure 4.10 it is possible to see the game field where the event is running. Since there were only six devices and three of them were needed to work as anchors, the tests made where with three moving tags like it can be seen in the picture. Since there is a possibility to assign how many

Figure 4.10: Event Running Window

tags will be placed in each time the system will then be able to draw either a blue or a red symbol on screen to help differentiate the players and better understand their movements.

In the end of the practice or game we have the possibility to export the details of what happened during it so that statistics can be formed around those details.

Below is an example of a line in the resulting CSV file from a test where the first value corresponds to the player name, the second and third to a 2D position in the field and the last one is the time at which that position was calculated so distances ran by a certain player can be discover with the difference in time and position in consecutive measurements.

```
Player 0, 2.23, -1.89, 30.05.2017 15:51:36.664
```

Chapter 5

Results

This chapter contains results of some tests made with the developed software to assess how it performs in real life and to assess what steps can be made in the future to fix any shortcomings the system currently might have.

5.1 Distance Measurement Test

First some simple distance tests where run to compare the functioning of the hardware used in chapter 3 and the one in chapter 4.

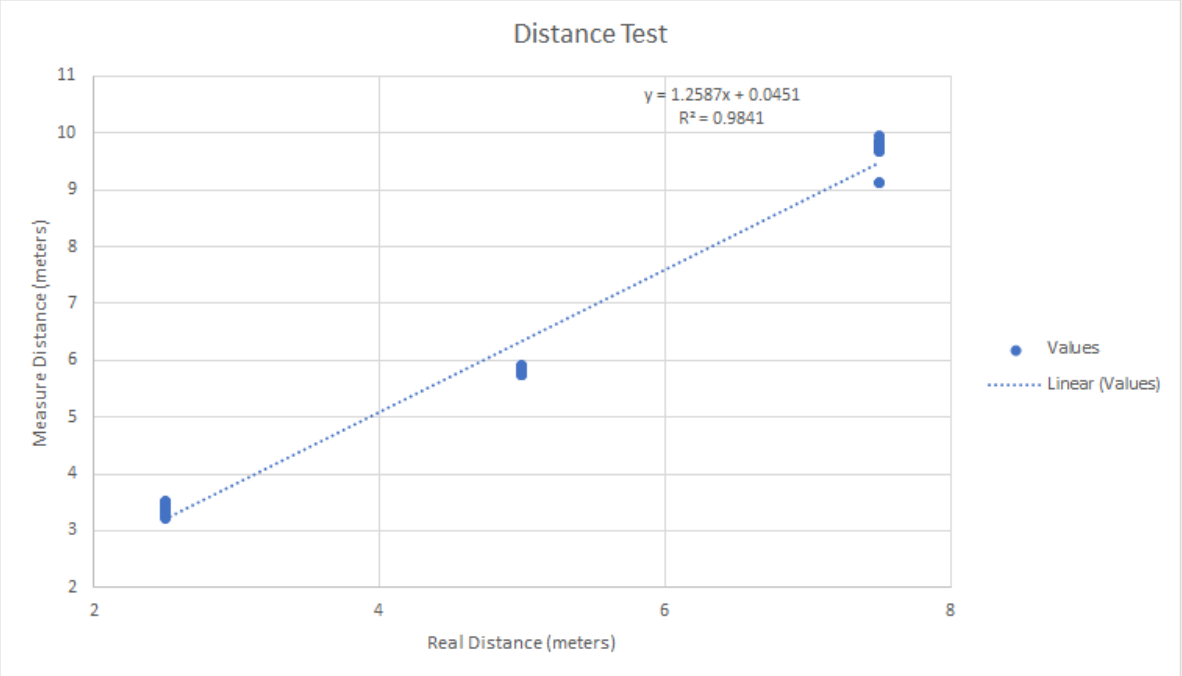


Figure 5.1: Distances' Graph

Three tags were placed at the distances of 2.5, 5 and 7.5 meters, in a place where there wasn't any disturbance or obstacles between the tags and the anchor collecting the values, as the system was left running for a few seconds to collect several distances to be analysed (those values can be more closely viewed in Appendix A).

By analysing the resulting graphic in the figure 5.1 and the graphic in the figure 3.4 we can conclude that the localino devices have slightly better accuracy than the ones used in the preliminary tests and as such the error calibration doesn't need to be as big.

5.2 Calibration Test

The calibration is an important part of the process of tracking the players inside the field as it gives the system the reference points needed to run the needed calculation.

As stated in section 4.3, the calibration processed is achieved by placing several devices in key positions, measuring their distances and then calculating their positions. For this test a small field was created with the measurements of 5.5 meters by 4 meters. A tag was placed in the centre with the other two placed on the long side, one in each corner. An anchor was positioned between them while the other two were placed on the other corners, slightly outside the field.

The collected distances led to the values presented in the table in figure 5.1 that were then used in the calibration process.

Table 5.1: Average Distances Between Devices (in meters)

	Tag 1	**Tag 2**	**Tag 3**	**Anchor 2**	**Anchor 3**
Anchor 1	3.77545	4.91377	7.15702	5.715	5.4986
Anchor 2	3.1524	6.6267	4.928	x	4.65226
Anchor 3	2.33281	3.00339	3.00188	x	x

The values represented in figure 5.1 resulted in 31 iterations of the calibration algorithm and some of those iterations can be viewed in the figures 5.2,5.3,5.4, 5.5 and 5.6 to better analyse how it is performing. The values in the figures represent the calculated distances in meters.

Figure 5.2 is the representation of the starting values giving to the algorithm so it knows the general direction on where the devices where placed and as such will keep the same orientation that will later be used to represent the field and the moving players.

In the first iteration (figure 5.3), the algorithm discovered values in the general vicinity of where the devices are positioned. By analysing figures 5.4 and 5.5 the algorithm quickly arrives at would be the final positions of the devices. The reason is that the minimization method used until then is based on the gradient descent and the steps in each iteration are more noticeable, specially in this order of magnitude. The remaining iterations are based more on the Gauss-Newton method so each update just changes the positions of the devices by a small margin around the fourth or fifth decimal place which isn't really relevant in the problem we are trying to solve.

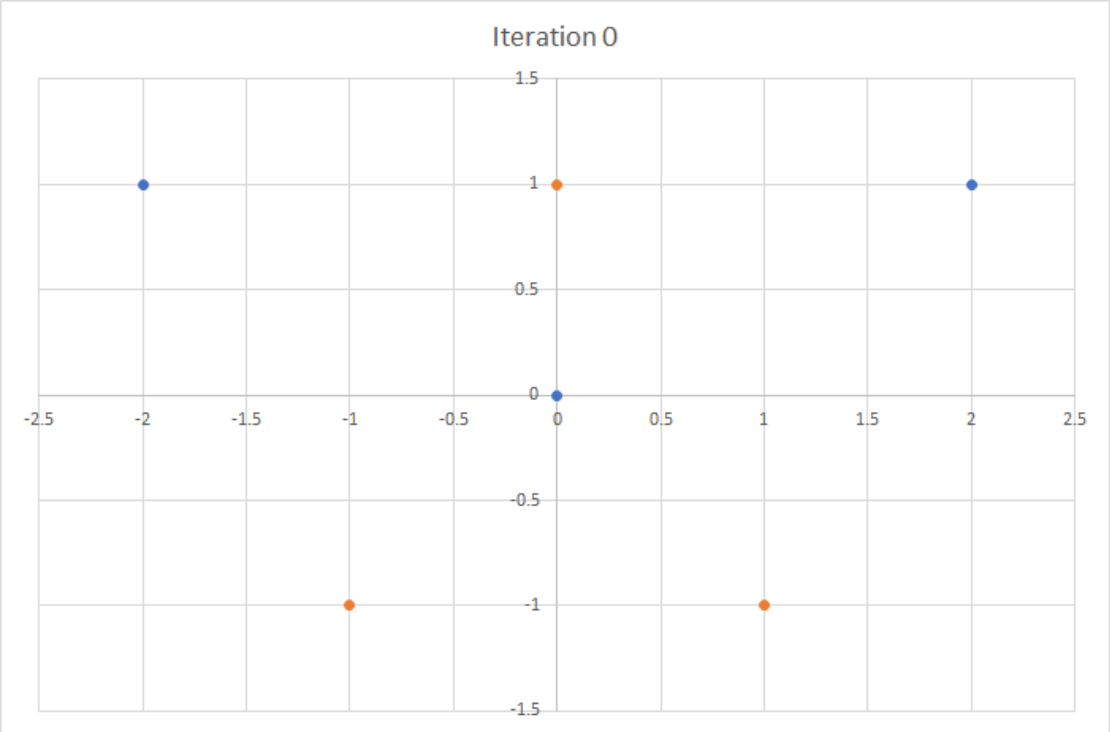


Figure 5.2: Iteration 0

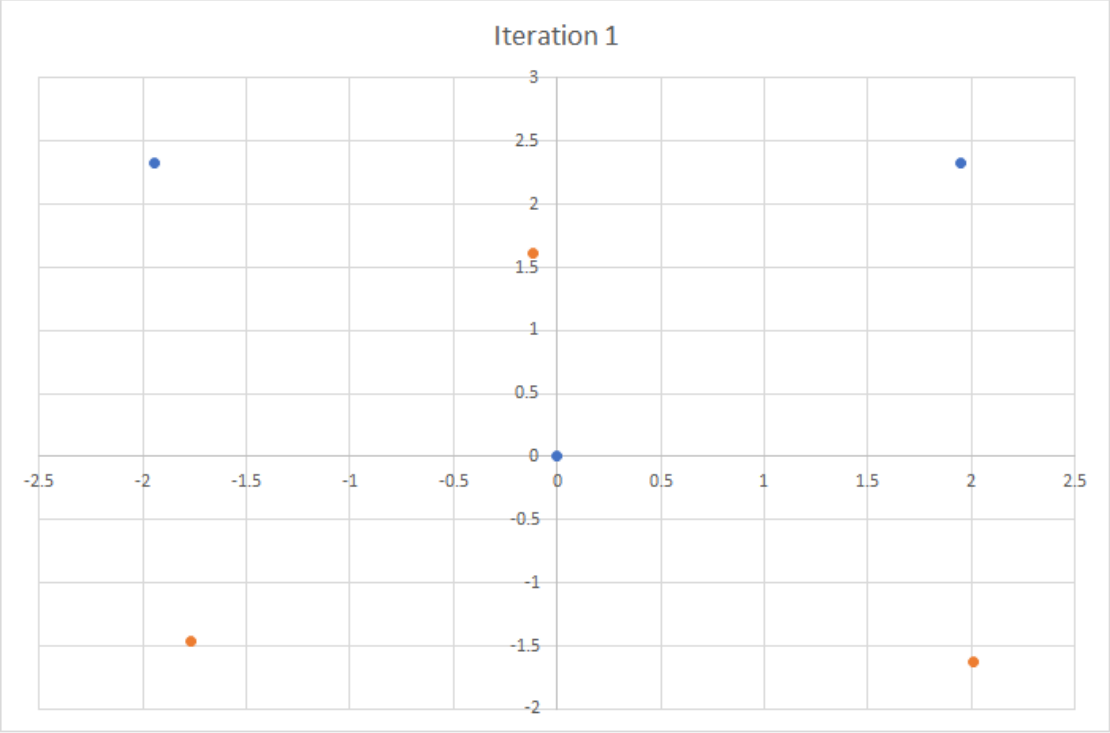


Figure 5.3: Iteration 1

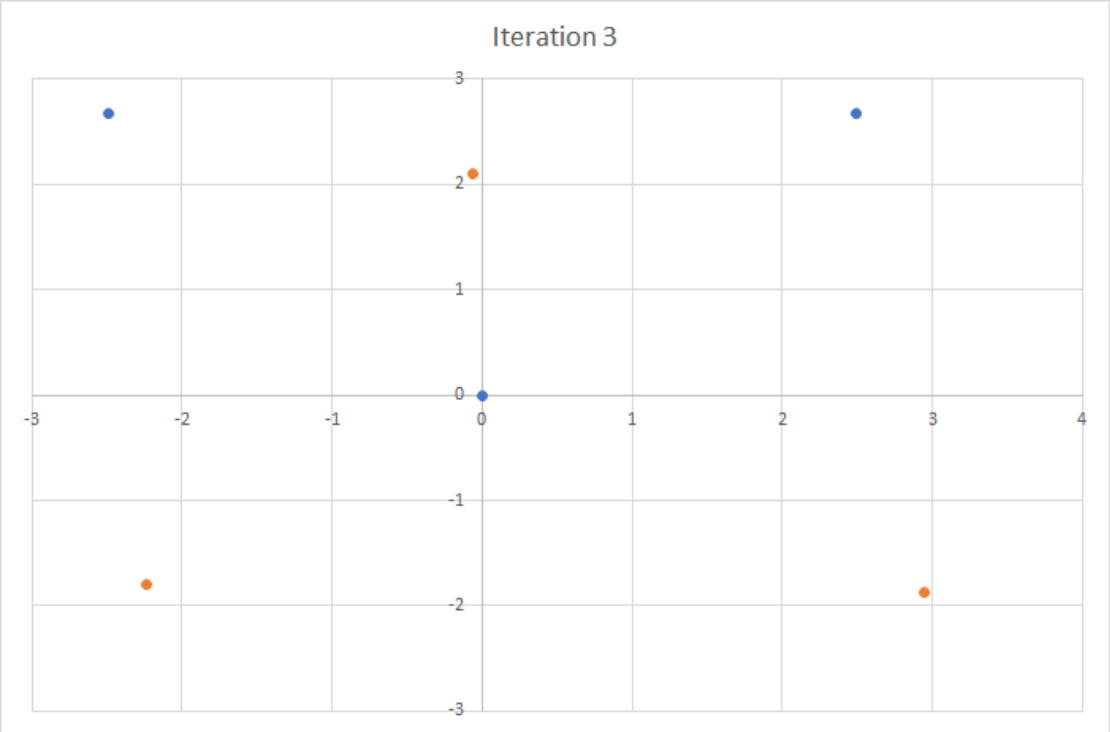


Figure 5.4: Iteration 3

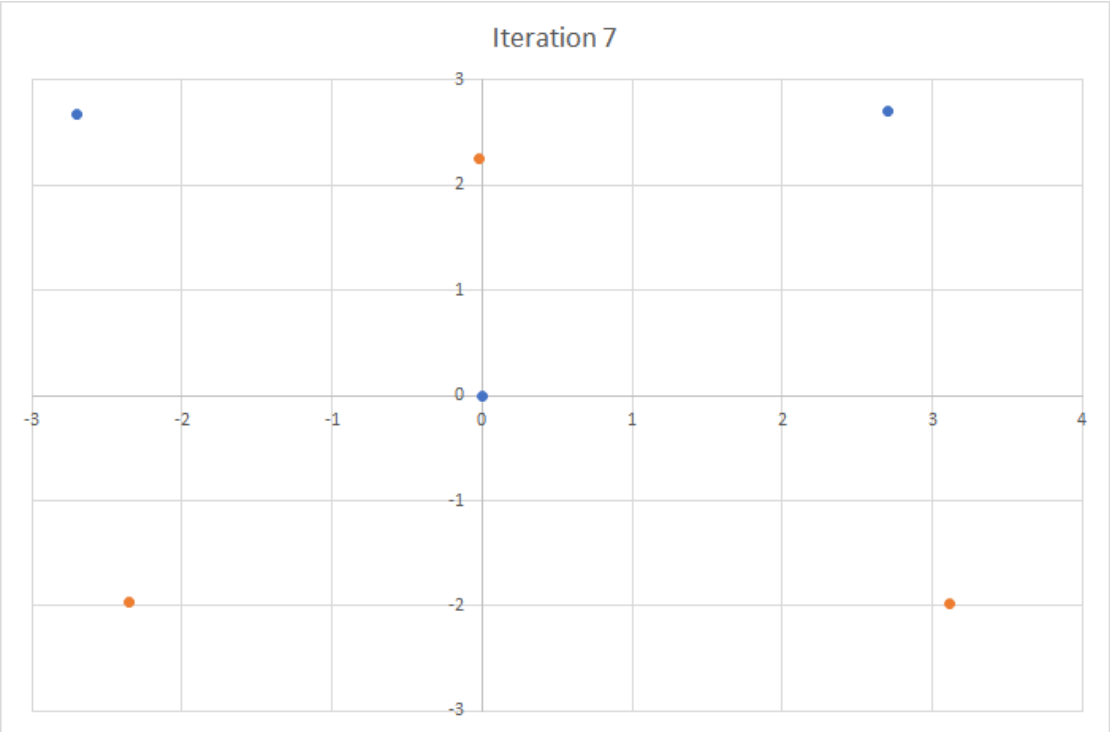


Figure 5.5: Iteration 7

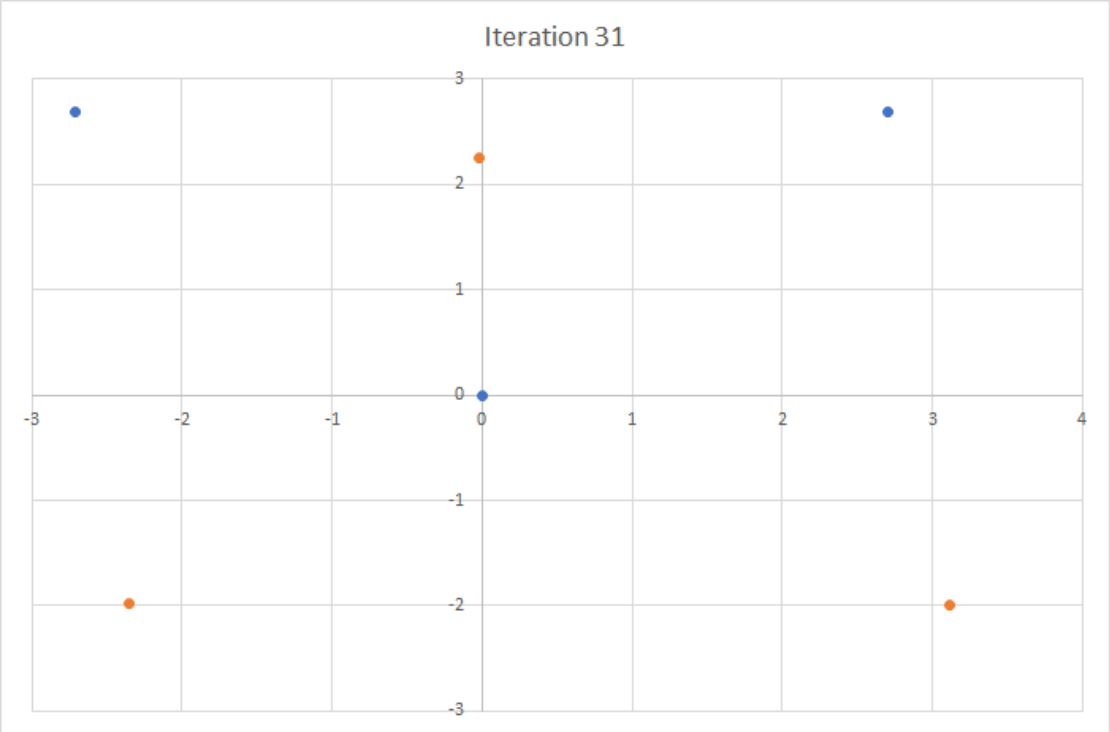


Figure 5.6: Iteration 31

The possible reason the values in the Y-axis are slightly bigger than expected maybe because of a slight miss-position of the centre tag. Any error in its position is duplicated by two as it will lead to a difference in the Y-axis is mirrored to the opposite side of the other tags. A way to fix this would be to add another tag in one of the remaining corners or perhaps even fill all corners with tags as any asymmetries in the field will more easily noticed in the calibration process.

Chapter 6

Conclusions and Future Work

6.1 Implemented Features

Currently, the implemented firmware is able to handle the communication and measurement of distances between either one anchor and one tag or up to four anchors and multiple tags, only limited by the number of physical devices. Those values can be all adjusted in the visualization software without the need to re-upload the firmware on to the devices every time a device is added to removed from the system.

The software used for visualization can be used in any sort of field as it possesses the ability to calibrate itself using the devices connected to the system and doesn't need to have previous knowledge about the type of field it currently is working on. The representation of the field is just to give the user a more familiar look and can be changed to other images, depending on the needs. All the values used either for calibration or for the players' position can be exported to a text file so they can be later analysed and used for any sort of debugging process.

6.2 Limitations

The first tests made with the localino devices in order to create the state machines needed to handle the communications between the different devices were made with distances two meters as there was a constant need to connect them to the computer to update their firmware. In those conditions the system behaved as expected and there were almost no messages lost and the visualization software was built based on those tests.

Because of that it was assumed the system would behave with similar results even if the distances between the devices would rise to the appropriate measurements of a sports' field. The problem was that when the system was finally tested with big enough distances, the number of messages lost was too great, as small disturbances in the space between the devices might lead to some part to the transmission to not be recognized by one of the devices.

Even with the messages lost, the system was still able to retrieve some successful transmissions and the results from them came with the same degree of error as if the devices where closer to each other but it wasn't consistent enough to accurately calibrate the overall system and then track the players fast enough for it to give a fluent movement on screen as it could take too long to retrieve a successful transmission.

6.3 Future Work

Due to the issues in the transmission of messages at longer distances, the first corrections needed to be made in the current system are the ones to the firmware running in the microprocessors. A different approach is needed in order to decrease the number of messages lost either by altering the way messages are being transmitted or by adding more devices in the system to reduce the distance between anchors and tags and create smaller zones inside the field similar to the work done in [LTLM13].

Another helpful addition to decrease the influence of messages lost when the event is already running is by adding a prediction algorithm alongside the already existing position calculation so it can give a more precise location for a player. Some human related parameters should also be added so the representation of the player doesn't move to a place in the field where it isn't humanly possible for it to happen because the distances measured had a degree of error that wasn't expected.

Lastly a web application would need to be created to import all the details of any event run and provide a user with several statistics about each player or about the team as a whole.

References

[Dar] Dart ultra wideband (uwb) technology. https://www.zebra.com/gb/en/solutions/location-solutions/enabling-technologies/dart-uwb.html. Accessed: 2017-01-24.

[DWM] Scensor dwm1000 module for developers of real time location and indoor positioning systems, location based services, internet of things and wireless sensor networks. http://www.decawave.com/sites/default/files/resources/DWM1000-Datasheet-V1.5.pdf. Accessed: 2017-01-24.

[Gav13] ©Henri P. Gavin. The levenberg-marquardt method for nonlinear least squares curve-fitting problems. 2013.

[HL] Heuel and Loeher. Localino, an open source indoor localization. from simple ranging application to full 3d positioning. https://www.localino.net/. Accessed: 2016-10-18.

[Hä16] Matti Hämäläinen. Chapter 1 - ieee802.15.4-2011 {WBAN}. In Matti Hämäläinen, editor, *Wireless {UWB} Body Area Networks*, pages 1 – 17. Academic Press, Oxford, 2016.

[Laz] Lazarus. Delphi compatible cross-platform ide for rapid application development. it has variety of components ready for use and a graphical form designer to easily create complex graphical user interfaces. https://www.lazarus-ide.org/. Accessed: 2017-03-26.

[LTL+09] Jia Liu, Xiaofeng Tong, Wenlong Li, Tao Wang, Yimin Zhang, and Hongqi Wang. Automatic player detection, labeling and tracking in broadcast soccer video. *Pattern Recognition Letters*, 30(2):103 – 113, 2009. Video-based Object and Event Analysis.

[LTLM13] Wei-Lwun Lu, Jo-Anne Ting, James J. Little, and Kevin P. Murphy. Learning to track and identify players from broadcast sports videos. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(7):1704–1716, 2013.

[Poz] Pozyx. centimeter positioning for arduino. https://www.pozyx.io/. Accessed: 2017-03-06.

[QT] QT. Cross-platform software development for embedded and desktop. https://www.qt.io/. Accessed: 2017-03-26.

[Swi] Swing. A lightweight java graphical user interface (gui) widget toolkit. http://docs.oracle.com/javase/tutorial/uiswing/. Accessed: 2017-03-26.

REFERENCES

[THU12] STALINBABU THUMMALAPALLI. Wi-fi indoor positioning. Master's thesis, Halmstad University, School of Information Science, Computer and Electrical Engineering (IDE), 2012.

[Tro] Thomas Trojer. A library that offers functionality to use decawave's dw1000 chips/-modules. https://github.com/thotro/arduino-dw1000.

[Way] Wayne. Wayne's tinkering page - uwb ranging with the decawave dwm1000. https://sites.google.com/site/wayneholder/uwb-ranging-with-the-decawave-dwm1000---part-ii.

[Wut] Joachim Wuttke. lmfit – a C library for Levenberg-Marquardt least-squares minimization and curve fitting. Version 6.1, retrieved on 03/06/2017 from http://apps.jcns.fz-juelich.de/lmfit.

Appendix A

Measurement Data

Table A.1: Distances Measured

Measured Distance	Real Distance	Measured Distance	Real Distance	Measured Distance	Real Distance
3.41	2.5	3.29	2.5	3.35	2.5
3.3	2.5	3.34	2.5	3.39	2.5
3.45	2.5	3.29	2.5	3.4	2.5
3.37	2.5	3.41	2.5	3.37	2.5
3.42	2.5	3.37	2.5	3.36	2.5
3.35	2.5	3.35	2.5	3.36	2.5
3.31	2.5	3.33	2.5	3.44	2.5
3.35	2.5	3.35	2.5	3.27	2.5
3.41	2.5	3.36	2.5	3.42	2.5
3.33	2.5	3.32	2.5	3.35	2.5
3.45	2.5	3.32	2.5	3.41	2.5
3.3	2.5	3.24	2.5	3.35	2.5
3.33	2.5	3.32	2.5	3.35	2.5
3.35	2.5	3.33	2.5	3.43	2.5
3.31	2.5	3.34	2.5	3.37	2.5
3.29	2.5	3.4	2.5	3.32	2.5
3.42	2.5	3.38	2.5	3.38	2.5
3.34	2.5	3.36	2.5	3.34	2.5
3.3	2.5	3.34	2.5	3.33	2.5
3.35	2.5	3.35	2.5	3.38	2.5
3.38	2.5	3.34	2.5	3.36	2.5
3.47	2.5	3.28	2.5	3.46	2.5
3.35	2.5	3.3	2.5	3.38	2.5
3.31	2.5	3.31	2.5	3.46	2.5
3.4	2.5	3.37	2.5	3.36	2.5
3.46	2.5	3.4	2.5	3.32	2.5
3.33	2.5	3.23	2.5	3.29	2.5
3.52	2.5	3.34	2.5	3.33	2.5
3.4	2.5	3.28	2.5	3.35	2.5
3.51	2.5	3.31	2.5	3.32	2.5
3.33	2.5	3.39	2.5	3.27	2.5
3.33	2.5	3.45	2.5	3.34	2.5
3.4	2.5	3.34	2.5	3.34	2.5
3.38	2.5	3.36	2.5	3.45	2.5
3.48	2.5	3.38	2.5	3.37	2.5
3.4	2.5	3.31	2.5	3.39	2.5
3.34	2.5	3.33	2.5	3.41	2.5
3.28	2.5	3.33	2.5	3.3	2.5
3.33	2.5	3.35	2.5	3.31	2.5
3.26	2.5	3.26	2.5	3.33	2.5
3.26	2.5	3.27	2.5		

Table A.2: Distances Measured Cont.

Measured Distance	Real Distance
5.78	5
5.8	5
5.83	5
5.77	5
5.86	5
5.81	5
5.85	5
5.77	5
5.76	5
5.84	5
5.85	5
5.83	5
5.83	5
5.79	5
5.79	5
5.83	5
5.88	5
5.85	5
5.81	5
5.77	5
5.82	5
5.8	5
5.82	5
5.83	5
5.8	5
5.86	5
5.77	5
5.8	5
5.78	5
5.85	5
5.86	5
5.88	5
5.78	5
5.77	5
5.81	5
5.83	5
5.86	5
5.92	5

Measured Distance	Real Distance
5.85	5
5.83	5
5.81	5
5.85	5
5.84	5
5.79	5
5.85	5
5.81	5
5.81	5
5.86	5
5.86	5
5.82	5
5.84	5
5.8	5
5.78	5
5.87	5
5.87	5
5.82	5
5.81	5
5.91	5
5.86	5
5.8	5
5.82	5
5.8	5
5.8	5
5.82	5
5.85	5
5.8	5
5.85	5
5.79	5
5.75	5
5.77	5
5.86	5
5.85	5
5.79	5
5.85	5
5.74	5
5.8	5
5.83	5

Table A.3: Distances Measured Cont.

Measured Distance	**Real Distance**	**Measured Distance**	**Real Distance**
9.76	7.5	9.77	7.5
9.95	7.5	9.77	7.5
9.78	7.5	9.72	7.5
9.74	7.5	9.68	7.5
9.72	7.5	9.76	7.5
9.7	7.5	9.79	7.5
9.73	7.5	9.73	7.5
9.77	7.5	9.8	7.5
9.73	7.5	9.76	7.5
9.72	7.5	9.74	7.5
9.8	7.5	9.8	7.5
9.81	7.5	9.75	7.5
9.85	7.5	9.71	7.5
9.77	7.5	9.75	7.5
9.72	7.5	9.77	7.5
9.81	7.5	9.77	7.5
9.8	7.5	9.79	7.5
9.76	7.5	9.74	7.5
9.74	7.5	9.74	7.5
9.74	7.5	9.79	7.5
9.8	7.5	9.78	7.5
9.77	7.5	9.71	7.5
9.76	7.5	9.77	7.5
9.12	7.5	9.72	7.5
9.74	7.5	9.69	7.5
9.86	7.5	9.77	7.5
9.78	7.5	9.76	7.5
9.81	7.5	9.8	7.5
9.83	7.5	9.77	7.5
9.13	7.5	9.73	7.5
9.82	7.5	9.84	7.5
9.79	7.5	9.75	7.5
9.78	7.5	9.72	7.5
9.73	7.5	9.71	7.5
9.7	7.5	9.8	7.5
9.72	7.5	9.75	7.5
9.72	7.5	9.84	7.5
9.73	7.5	9.73	7.5